A Local Information Based Distributed Algorithm Constructing 3-Nodes Rectilinear Steiner Tree in Virtual Grid Networks

Yonghwan Kim† Fukuhito Ooshita† Hirotsugu Kakugawa† Toshimitsu Masuzawa†

†Graduate School of Information and Technology, Osaka University, Osaka, Japan

Abstract

In this paper, we present a local information based distributed algorithm which constructs a rectilinear steiner tree among 3 nodes in virtual grid networks. A virtual grid network is obtained by virtually dividing a wireless network into a grid of geographical square regions called *cells*. A single node is selected as a *router* at each cell and inter-cell communication is realized by using the routers. Other nodes in the cell have no responsibility for inter-cell communication and can become inactive to save energy consumption.

We suppose one special node (named a *home* node) and several moving nodes (named *target* nodes) in virtual grid networks. And we consider maintenance of inter-cell communication paths to each target node from the a *home* node. In this paper, we propose an optimizing protocol in virtual grid networks, which can transform arbitrary given set of paths (from a home node to each target node) to rectilinear steiner tree in order to save energy consumption.

1 Introduction

Recently, wireless networks, such as MANETs (Mobile Ad-hoc NETworks) or WSNs (Wireless Sensor Networks), become popular and important in the distributed systems. In the wireless networks, nodes are deployed on a two-dimensional plane, and each node can directly communicate only with nodes within its communication range. If the destination node (the node receives the message) is outside of the communication range of the source node (the node sends the message), the message should be relayed to the destination node.

The topology of wireless networks can be changed frequently because of the mobility property of a node or node failures. Moreover, each node in the wireless networks has resource scarcity, for example, processing power, energy, and storage. Therefore, the key issues of the wireless routing protocols contain adaptability to the network dynamics and reduction of resource consumption[1, 2].

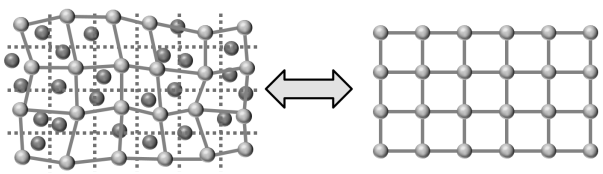

Figure 1: A Virtual Grid Network

Fig. 1 represent a virtual grid network[3], which is obtained by virtually dividing a wireless network into a grid of geographical square regions called *cells* of the same size. The size of the cells is determined so that any nodes in the same cell or in the neighboring cells can directly communicate with each other.

A single node is selected as a *router* at each cell and inter-cell communication is realized by using the routers. Other nodes in the cell have no responsibility for inter-cell communication and can become inactive (e.g., sleep) to save energy consumption. Since energy efficiency is one of the most critical issues in wireless networks, a router of each cell should be periodically reselected.

We suppose one special node (named a *home* node) and several moving nodes (named *target* nodes) in virtual grid networks. And we assume that the set of paths, which consists of each path to each target node from a home node, are given. This implies that if there are T target nodes in the virtual grid network, T paths (from a home node to each target node) are given.

In this paper, we consider maintenance of inter-cell communication paths to each target node from the node. As we mentioned above, the power consumption is one of the important issue in wireless sensor networks, therefore, we consider that our goal is the construction of a rectilinear steiner tree (means the steiner tree on the grid topology) which connects a home node to all target nodes. Steiner tree ensures that the total number of edges between routers from its definition. However, construction of rectilinear steiner tree in grid networks is known as *NP-hard* problem[4]. This implies that the devising of an distributed algorithm, which can find an optimal solution with local information only, is nearly impossible. Thus, in this paper, we consider only three nodes, two target nodes and one home node, in a virtual grid network. We propose an optimizing protocol in virtual grid networks, which can transform arbitrary given two paths (from a home node to two target nodes) to rectilinear steiner tree for saving energy consumption.

2 3-nodes Rectilinear Steiner Tree Problem

In this Section, we define 3-nodes rectilinear steiner tree (RST) problem in detail, and introduce some characteristics of this problem.

We suppose one special node (named a *home* node) and two moving nodes (named a *target* nodes) in virtual grid

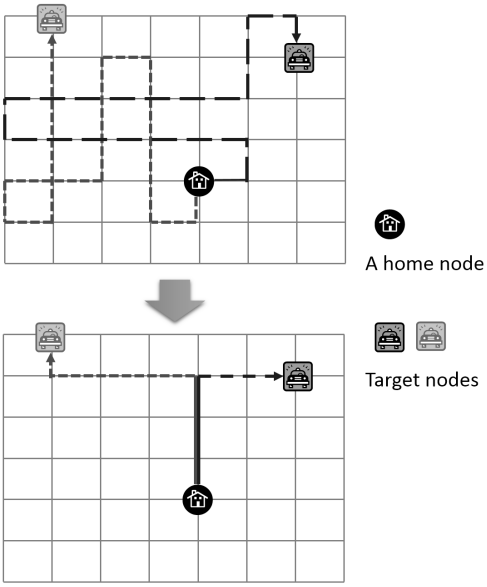

Figure 2: An example of 3-nodes RST problem

networks. And we assume that two arbitrary paths, which are the paths to two target nodes from a home node, are given. The goal of our protocol is the construction of the minimum rectilinear steiner tree among these three nodes. Fig. 2 illustrates an example of 3-nodes RST problem. Initially, two paths between each target node and a home node are given. Certainly, these path are not the shortest path between them and can be updated by the moving of a target node. Our proposed protocol construct the minimum rectilinear steiner tree like the lower part of Fig. 2 using local information and local updates only.

3-nodes RST problem can be changed to shortest-path tree problem. This implies that if the construction of a shortest-path tree with minimum total edge weight (in the case of a virtual grid network, the minimum number of edges) is available, that tree also becomes a rectilinear steiner tree. This can be allowed when there are only 3 nodes in a virtual grid network.

To help to understand, we use the cartesian coordinate plane (two-dimension) and we suppose a home node is exist on origin (0,0) of this plane. And we can represent the positions of two target nodes as a pair of x and y value. Fig.

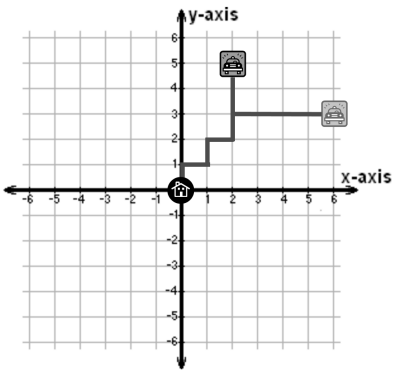

Figure 3: A steiner tree when two target nodes are in the same quadrant

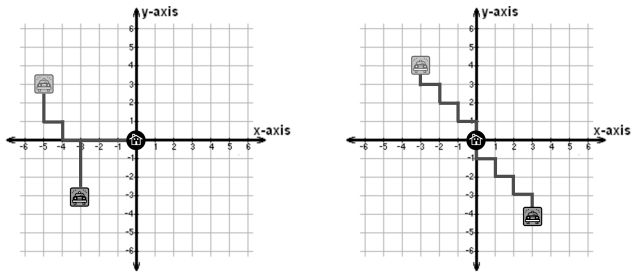

Figure 4: The other exmaples of steiner trees

3 represent an example of a steiner tree when two target nodes are in the same quadrant of a cartesian coordinates. In this case, we can simply construct a steiner tree as follow steps (if we can know global information).

1. Make shortest path to $(min(|x_1|, |x_2|), min(|y_1|, |y_2|))$ from a home node $(0, 0)$. (where two target nodes are located on (x_1, y_1) and (x_2, y_2)).
2. Make shortest (becomes straight line) path to each target node from $(min(|x_1|, |x_2|), min(|y_1|, |y_2|))$.

Likewise, we can easily construct a steiner tree in other two cases: two target nodes are located in the neighboring quadrants, and in the opposite quadrants. Fig. 4 represent an example of a steiner tree in the above two cases. Note that each (spanning) trees in Fig. 3 and 4 becomes not only a shortest-path tree but a steiner tree.

Our protocol adopts the routing protocol *Zigzag*[5], which can transform any given inter-cell path to a shortest (or minimum-hop) one by repeatedly applying local updates on the path (will be introduced in Section 3). In our protocol, we construct a shortest path to each target node from a home node using the protocol *Zigzag*, and, if possible, we combine some parts of two paths to reduce the number of edges. Our protocol can be operated on each router locally and adaptively (unaffected by target nodes' moves) in a virtual grid network using local information only.

3 Our proposed protocol

As we mentioned, we adopt a routing protocol named *Zigzag* to make a shortest path to each target node from a home node, and we propose some new protocols to combine some parts of two paths for reduction the number of edges contained in tree.

3.1 A routing protocol *Zigzag* and its modification

Zigzag is a local-information-based self-optimizing routing protocol in virtual grid networks. Protocol *Zigzag* find a shortest path between two nodes by repeatedly applying local updates to the path until it converges to a shortest path. *Zigzag* detects a redundancy of the recent path locally with making *zigzag-based* path.

Zigzag defines only three local updates on the node p_i as Fig. 5.

We consider the combining of two paths which are transformed (or been transforming) by *Zigzag*. However, the

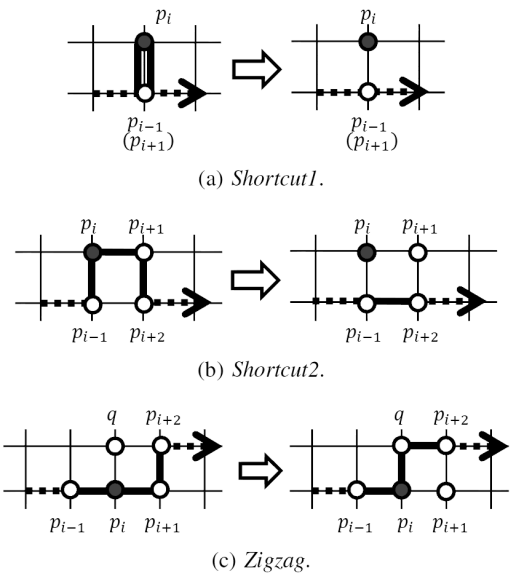

Figure 5: Three local updates to a path in *Zigzag*

converged shortest path can be different depending on the initial path, even if the positional relation between two nodes is exactly same. Fig. 6 represent two possibilities of the converged shortest path by *Zigzag*. When a target node is located in the first quadrant, a shortest path can be the one of two possible paths after convergence. In the case of Fig. 6, a shortest path can be started with up direction or right direction. This causes some difficulties when we find the relation between two paths. For example, if two target nodes are in the same quadrant, the overlapping of some parts of two paths can be expected. However, two converged paths can be completely different depending on starting directions.

Therefore, we modify the protocol *Zigzag* slightly. If a home node finds some specific starting directions, a home node updates those directions. Table 1 shows the detail rules of starting directions.

This modification makes it an open possibility to combine two paths. We introduce our protocol in the Section 3.2.

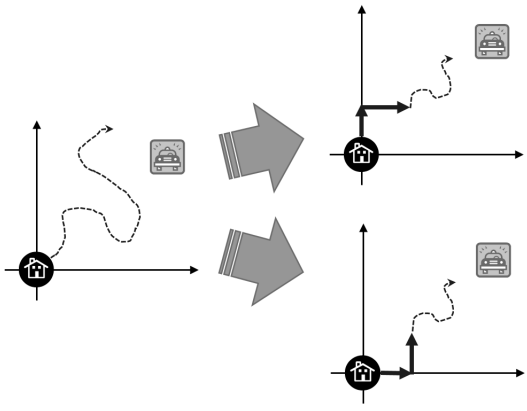

Figure 6: Two possible converged paths by *Zigzag*

Table 1: Rule for fixing the starting directions

2-hop directions from home	modified directions
{UP, RIGHT}(1st quadrant)	{RIGHT, UP}
{LEFT, UP}(2nd quadrant)	{UP, LEFT}
{DOWN, LEFT}(3rd quadrant)	{LEFT DOWN}
{RIGHT, DOWN}(4th quadrant)	{DOWN, RIGHT}

3.2 Path Combining Protocol

In this Section, we introduce our protocol to reduce the number of edges combining two paths. In the previous Section, we modified the protocol *Zigzag* thus we can easily find the positional relationship between two paths. We explain how to combine two paths in detail.

3.2.1 Two target nodes are in the neighboring quadrants

At first, we consider the case when two target nodes are in the neighboring quadrants (e.g., one target node is in the 1st quadrant, and another one is in the 2nd quadrant).

Fig. 7(a) shows an example case when two target nodes are in the neighboring quadrants, 1st and 2nd quadrants. In this case, we can find *U-shaped* path, from $(0, 1)$ to $(1, 1)$, on a home node (the origin). From this U-shaped path, a home node recognizes that two target nodes are in the neighboring quadrants and two paths can be combined. Note that, this recognition might be incorrect because *Zigzag* is not converged yet, but a home node can decide it at the current moment and our protocol can resolve this local miss.

Fig. 7(b) presents the situation after combining detected U-shaped path. Total number of edges is less than Fig. 7(a), however we cannnot find more combining points although this is not optimal solution. Therefore, we introduce a new virtual node named *virtual home* node (*vHome*). *vHome* is located on the home node initially, and after combining two path on U-shaped path as Fig. 7(b), *vHome* moves one hop along the combined path. In the case of Fig. 7(b), *vHome* is located on $(0, 1)$. *vHome* operates exactly same as a home node, this changes the *Zigzag* paths on Fig. 7(b). The path to left target changes its starting directions {UP, LEFT} due to the modification of *Zigzag* (Section 3.1). The zigzag part of the path to right target moves left by *Zigzag* protocol.

Fig. 8(a) illustrates the path after *vHome* is located on $(0, 1)$. Because of local updates of *Zigzag* protocol and its modification, our protocol can find the combining point again. A virtual home moves repeatedly and this protocol can be eventually converged as Fig. 8(b).

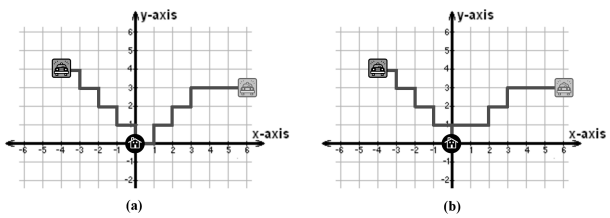

Figure 7: An example case of neighboring quadrants

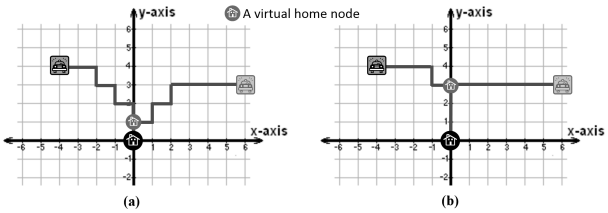

Figure 8: Coordinating using a virtual home

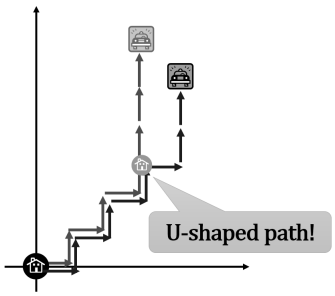

Figure 9: When two target nodes are in the 1st quadrant

3.2.2 Two target nodes are in the same quadrants

In the case when two target nodes are in the same quadrants (e.g., both of them are located in the third quadrant), some of paths are overlapped because of *Zigzag* protocol and its modification.

Fig. 9 shows an example when both of two target nodes are located in the first quadrant. We can easily find the overlapped (common) path from a home node (to ($min(|x_1|, |x_2|)$, $min(|y_1|, |y_2|)$). In this case, *vHome* can move along this common path and we can find a new U-shaped path (if exists) for combining two paths as we explained in Section 3.2.1.

3.2.3 Two target nodes are in the opposite quadrants

If two target nodes are in the opposite quadrants (e.g., one is in the 1st quadrant and another is in the 3rd quadrant), there is no U-shaped path or common path. Therefore, any combining of paths are never occurred.

3.2.4 Supplement of our protocol

Our protocol using *vHome* basically updates the path, if it finds U-shaped path. However, we cannnot know whether a protocol *Zigzag* is converged or not using local information only. As we mentioned in Section 3.2.1, our protocol's local update (combining) is sometimes incorrect. However, our protocol allows not only moving forward but also moving backward of *vHome* when it finds U-shaped path.

Fig. 10 shows the case that *vHome* moves backward (toward a home node) because of detecting U-shaped path. At this moment, *vHome* is located on $(0, 3)$ because of detecting miss (this might be occurred when *Zigzag* protocol is not converged yet, or a target node moves after combining). However, in this case, *vHome* moves backward repeatedly, and our protocol is converged when *vHome* is located on $(0, 1)$. Even if *vHome* is moved by detecting

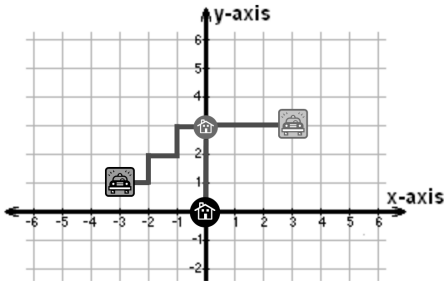

Figure 10: The case that *vHome* moves backward

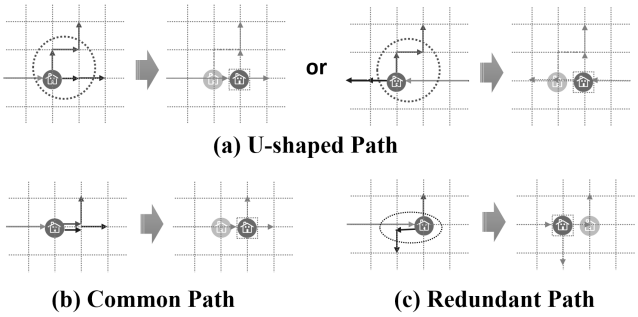

Figure 11: Three rules of our protocol

some common path as we introduced in Section 3.2.2., it can be move backward due to detecting of U-shaped path.

Finally we introduce one more simple rule to our protocol. By our protocol, the path to *vHome* from a home node is created. When *vHome* is not on the origin, if the direction just after *vHome* is the reverse direction of one just before *vHome*, *vHome* moves backward with one hop because the last hop of common path is redundant trivially.

Fig. 11 summarizes the local update's rule of our protocol.

3.2.5 A Problem of our proposed protocol and its solution

In this Section, we introduce a problem of our protocol and its solution. Fig. 12 shows an example of our protocol's incorrect convergence. The difference of convergence time between two protocol, our proposed protocol and *Zigzag* causes this incorrect convergence. However our proposed protocol is operated on each node locally, thus all nodes never recognize this miss convergence problem.

We can resolve this problem easily adopting *Zigzag* pro-

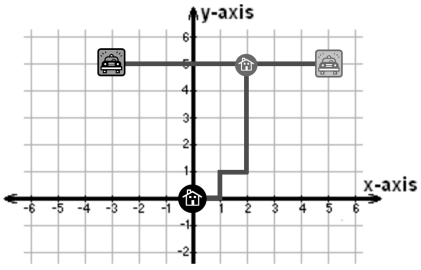

Figure 12: Incorrect convergence of our protocol

tocol between a home node and *vHome*. The important point of this adopting is the direction of *Zigzag* protocol. In this *Zigzag* protocol, *vHome* is resposible for a source node and a home node becomes a destination node. In the case of Fig. 12, if *vHome* adopts *Zigzag* toward a home node, we can find U-shaped path on *vHome* because the *zigzag-path* is created near *vHome* like $\{(2,5),(2,4),(1,4),(1,3)...\}$.

4 Summary and Future Works

In this paper, we proposed a protocol using local information only, which can construct 3-node rectilinear steiner tree. Our protocol uses a protocol *Zigzag* to make a shortest path to each target node from a home node. We modify *Zigzag* protocol slightly, and this enables the detecting some paths which can be combined. We introduce the notion of a virtual home node and some local update rules for combining path. We also explain a problem of our protocol and its simple solution.

However, in order to realize the correct convergence, our protocol assumes that a virtual home node is located in correct position and the path to a virtual node home from a home node correctly. Therefore, we consider the modification of our protocol to determine a virtual home node with only local information.

Moreover, we has to prove that our protocol creates the optimal steiner tree. Before a theoretical proof, we implement a simulator of our protocol, and evaluate our protocol on the various size of virtual grid networks. As a result of our simulator, we can find our protocol is operated correctly.

Finally, we should analyze the convergence time of our protocol.

References

[1] S. Giordano, Mobile Ad-Hoc Networks. Wiley, 2000.

[2] J. Zheng and A. Jamalipour, Wireless Sensor Networks : A Networking Perspective. Wiley-IEEE Press, 2009.

[3] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed energy conservation for ad hoc routing,” in Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom ’01), 2001, pp. 70—84.

[4] F. K. Hwang, “On Steiner Minimal Trees with Rectilinear Distance”, SIAM Journal on Applied Mathematics, Vol.30, No.1, pp.104-114, 1976.

[5] S. Takatsu, F. Ooshita, H. Kakugawa, and T. Masuzawa, ”Zigzag: Local-information-based self-optimizing routing in virtual grid networks,” Proceedings of the 33rd International Conference on Distributed Computing Systems (ICDCS) , pp. 358-368, July 2013.